Underwater Image Processing using Graphics Processing Unit (GPU)

Sayooj Ottapura[1,*], *Rahul* Mistry[2], *Jatin* Keni[3], and *Chaitanya* Jage[4]

[1]Department of Electronics Engineering, Ramrao Adik Institute of Technology, Nerul, India
[2]Department of Electronics Engineering, Ramrao Adik Institute of Technology, Nerul, India
[3]Department of Electronics Engineering, Ramrao Adik Institute of Technology, Nerul, India
[4]Department of Electronics Engineering, Ramrao Adik Institute of Technology, Nerul, India

Abstract. Image processing is a method used for enhancement of an image or to extract some useful information from the image. It is a type of signal processing in which input is an image and output may be an image or any characteristics/features associated with that image. In this paper we will be focusing on a specific type of Image Processing i.e. Underwater Image Processing. Underwater Image Processing has always faced the problem of imbalance in colour distribution and this problem can be tackled by the simplest algorithm for colour balancing. We will be proceeding with the assumption that the highest values of R, G, B observed in the image corresponds to white and the lowest values corresponds to darkness. The underwater images are majorly saturated by blue colour because of its short wavelength and in this paper, we aim to enhance the image. We proposed a colour balancing algorithm for normalizing the image. The entire process will first be carried out on a CPU followed by a GPU. We will then compare the speedup obtained. Speedup is an important parameter in the field on image processing since a better speedup can help reduce the computation time significantly while maintaining a higher efficiency.

1 Introduction

Underwater image processing has always faced numerous challenges. Physical properties of the medium i.e. impurities present in water, cause deterioration effects which are absent in the pictures taken in air. Underwater images are primarily distinguished by their low visibility since light gets extremely attenuated as it advances in the water and hence the images taken result to have poor contrast and have a presence of haze. The two main difficulties faced are absorption and scattering. In absorption, specific frequencies of light get absorbed at different depths[1]. Since blue colour has the shortest wavelength it travels the farthest, making ambient underwater environments to be dominated by a bluish hue. Scattering is any divergence from a straight-line path[2]. Scattering occurs because of the presence of suspended particles in the water and in turn drops the visibility and contrast of the captured image. Light attenuation decreases the distance of visibility in clear water by about 20 meters and in turbid water by 5 meters or less[3]. The extent of effects of absorption and scattering is decided by the number of floating particles present in the water known as “marine snow”.

Underwater Image Processing can be done using either Hardware techniques or through Software techniques. The software image processing can be approached from two different perspectives: image restoration technique or as an image enhancement method.

i) The purpose of image restoration is to recover the original image from a degraded image by using models of both the images[4].

ii) Image enhancement focuses on certain criteria of the subject i.e. the image, and then produces a more visually appealing image. Image Enhancement method does not require any kind of physical model to process the image[4].

The approach that we will be discussing in the paper will be an Image enhancement method.

For the purpose of Image enhancement, we will be making use of Graphics Processing Unit (GPU). GPU is preferred over CPU because image processing algorithms typically consume a great deal of computer resources which can be easily provided by the numerous processing cores present in a GPU. Graphics processors in many technical aspects outperform other imaging acceleration methods, when compared to the fastest available Field-Programmable Gate Array (FPGA).

* Sayooj Ottapura: sayoojaugust98@gmail.com

2 Underwater Image Processing

Underwater imaging and image processing play significant roles in ocean science research. The basic purpose of the present work is to improve the quality of underwater images. A wide variety of methods are available for the purpose of image enhancement, each with its inherent advantages and disadvantages. Underwater pictures, as a natural phenomenon, have low contrast alongside showing a propensity for dominance of one colour and low sharpness.

In the year 2012, N. Shamsuddin; W.F. Wan Ahmad, B.B. Baharudin, M. Kushairi, M. Rajuddin and F. Mohd established a method on ”Significance level of image enhancement techniques for underwater images”. This work focused on diminishing colour[5]. In the very next year of 2013 M.S. Hitam, Wan Nural Jawahir Hj Wan Yussof, E.A. Awalludin and Z. Bachok operated on “Mixture contrast limited adaptive histogram equalization for underwater image enhancement”. Here they introduced a new method known as hybrid Contrast Limited Adaptive Histogram Equalization (CLAHE) colour spaces designed specifically for enhancement of the underwater image. The technique works CLAHE on colour spaces RGB and HSV. Tentative findings suggest that the potential solution substantially increases the picture quality of underwater photographs by improving contrast, and decreases noise and errors[6].

As mentioned earlier in Section 1, underwater image processing varies widely from visual image processing, mainly due to underwater channel dysfunctions such as absorption and scattering[7, 8]. Scattering can occur in two ways:

i) Forward scattering (light deviates while it travels from the object to the camera) leads to blurry images[9].

ii) Backward scattering (the water reflects a fraction of the light to the camera before it actually reaches the objects) usually limits the contrast of the images[9].

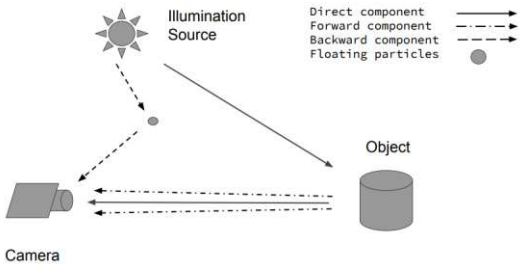

Fig. 1. The 3 elements of underwater image processing: i) the direct component (represented by a dash-dot line) ii) the forward component (represented by a straight line) and iii) the backward scatter component (represented by a dashed line)

As the images taken in underwater environment is mainly dominated by blue colour, we will be making use of a Colour Balancing Technique.

3 Image Enhancement using Graphics Processing Unit (GPU)

The Graphics Processing Unit (GPU) is widely used for real time graphics rendering in applications like high definition gaming, computer aided design software, etc[10]. GPU is becoming popular as cheap parallel supercomputer whose processing power can be utilized [11].

Even though original purpose of GPU is graphics rendering, in recent years the GPU is increasingly being used for scientific computations. GPUs are also used for general purpose computations like scientific computations, encryption/decryption etc. This type of GPUs is called as General-purpose GPUs (GPGPUs). Nowadays, GPU is used in video processing, image processing and in many more general-purpose works. Here we present an effective implementation of Simplest colour balance algorithm on the NVIDIA Tesla K80. Specifically, we showcase the effectiveness of our method by algorithm optimization. In result we display time relation between the implementation of the CPU and the GPU. We have run the algorithm using python and OpenCV on GPU[10].

Table 1. Specifications of Tesla K80.

Parameters	***Specification***
Maximum Double Precision Floating point precision	2.91 Téraflops
Maximum Single Precision Floating point precision	8.74 Téraflops
Memory Bandwidth(ECC Off)	480GB/sec
Memory size and Type	24GB GDDR5
CUDA cores	4992
Thermal solution	Passive Heatsink

Tesla K80 has two processing units. Each processing unit may be viewed as separate GPUs. So, this means Tesla K80 consists of 2 GPUs. Tesla K80 consists of 2 × 2496 cores and 2 × 12 Gigabytes of Video RAM. The operating frequency of each core is 562 MHz and the boost mode are 875 MHz[12, 13].

4 Design Implementation

In order to do the Image Enhancement process, we will be making use of the Simplest Colour Balancing (SCB) Algorithm. SCB is the Algorithm of choice because as mentioned earlier the images taken in underwater environment are majorly dominated by blue colour and so by using SCB we will be balancing out all the hues of RGB colour spectrum and thus reducing the dominance of blue making the resulting image to have an evenly balanced colour spectrum. The entire code for the algorithm is written and implemented in the Python Programming Language. The different libraries that are

required in the image processing are taken from Open Computer Vision (OpenCV).

4.1 Simplest Colour Balance (SCB) Algorithm

Colour balance refers to the removal of an inherent colour bias from an image. It means balancing the intensities of each channel separately. For instance, if an image appears to be too red, it is said to have a red hue over it. Removing the red cast returns the picture to balance. Colour casts can arise from many different causes. In underwater imaging, the images are usually blue casted. There are various ways to make an image more readable and removing the colour cast which is usually blue in underwater images. Herein we propose a Simplest Colour Balance algorithm which improves readability. The SCB Algorithm does not necessarily improve the image quality. The goal here is to scale the pixel values to a range of 0 to 255[14]. However, many underwater images pixels already occupy the 0 and 255 values. To obtain a better colour improvement, a small percentage of pixels having value equal to 255 and small percentage of pixels having value equal to 0 are clipped[14]. Clipping here means setting up the quantiles, V_{max} and V_{min}. V_{min} and V_{max} are calculated by using $N \times S_1/100$ and $N \times (1 - S_2)/100 - 1$ respectively[14]. The quantiles are defined so that small proportion of pixels get the value of the quantiles. Finding the quantiles can be done by using sorting algorithm or by using histogram algorithm.

In this algorithm we have used the sorting algorithm for finding the quantiles and then normalizing it to a normal 8-bit image range. An RGB image is of 24-bit. Each channel of an image is of 8-bit. In our algorithm we are applying normalization to each channel separately. Each channel pixel values will be sorting in ascending order. Depending on the user defined saturation percentage V_{max} and V_{min} are calculated. From the left side of an original image histogram S_1% of pixels will be saturated and from the right side of the histogram that is, the pixels having a value closer to 255 are saturated to S_2%. For example, S_1=0 means there will be no pixel saturation at the beginning. S_2=3 means there will be at most $N \times S_2/100$ at the end of histogram. We cannot guarantee that exactly $N \times (S_1 + S_2)/100$ pixels get saturated because the distribution of pixel value is discrete. After a certain percent of pixel saturation is done, the image is then normalized to a specific range. Normalization is nothing but the re-scaling of real valued numeric attributes into the range between 0 and 1 in general[15]. But here the image is 8-bit so the pixel value ranges from 0 to 255. So, normalization of pixels is done for a range of 0 to 255 using the formula mentioned below[14]:

$$f(x)=[(x - Vmin)\times(max - min)]/[(Vmax - Vmin)+min] \quad (1)$$

For every pixel x, $f(x)$ will be calculated. Here 'max' and 'min' are the range of normalization. The image is scaled to [min, max].

Let's take a look at an example, consider an image having a dimension of 183 rows and 275 columns. The total number of pixels are obtained by taking product of the number of rows and number of columns. Thus, for rows and columns and hence the number of pixels is 50325 which is usually represented as 'N'. S is the saturation value which we have considered to be 2 in this example. The image will be first separated with respect to each colour channel R, G and B. Thereafter we will employ the V_{max} and V_{min} equation for each channel. Let's now calculate the value of V_{min} and V_{max} by selecting $S_1 = S_2 = 2$ and N = 50325.

$$V_{min} = (50325 \times 2) / 200 = 503 \quad (2)$$

$$V_{max} = \{[50325 \times (1 - 2)] / 200\} - 1 = 49821 \quad (3)$$

Now by applying the Python indexing logic we get to know that the pixel value at the index position of 503 and 49281 of the blue channel arrays is 73 and 187 respectively. Hence, the final value of V_{min} comes out to be 73 and the value of V_{max} is 187 for the blue colour channel. So, the pixel values of pixel below the 503rd pixel will be replaced by V_{min} and the pixel values of pixels above 49821st will be replaced by V_{max} for the blue colour channel. Similarly, for green channel, the value of V_{min} is 73 and the value of V_{max} is 214. The pixel values of pixel below the 503rd pixel will be replaced by V_{min} and the pixel values of pixels above 49821st will be replaced by V_{max} of the green channel. For red colour channel, the V_{min} obtained is 16 and the V_{max} value is 116. The pixel values of pixel below the 503rd pixel will be replaced by V_{min} and the pixel values of pixels above 49821st will be replaced by V_{max} for the red channel. The last step is the normalization of the processed image having a range of b = [73,187], g = [73,214] and r = [16,116]. Each channel's pixel value range will be scaled between 0 to 255 and lastly the channel will be merged.

Sorting of N pixels require $O(N \times \log(N))$ operations and a temporary duplication of the same N pixels[14].

4.2 Python Programming Language

Python is a high-level programming language. Python is object-oriented with integrated dynamic semantics for web and app development. Since Python is simple and has fairly simple and easy syntax and also supports a large number of packages and modules it is the programming language of choice for our project. The Python version used in our codes is Python 3.5. We have made use of some popular Python modules such as NumPy, math, sys, time, etc.

4.3 Open Computer Vision (OpenCV)

OpenCV-Python is an open-source library basically used for computer vision assessment. Even though C and C++ are faster when compared to Python, Python is used in complex computations because of its simplicity. Another important attribute of Python is that it can be extended to

C and C++. This functionality allows us to implement complex codes in C or C++ and to build a Python wrapper which can then be used as Python modules. This holds two benefits for us:

i) Our program doesn't compromise on speed (because in actuality the C++ code is what is running in the background).

ii) Programming in Python comparatively easy.

NumPy support in OpenCV makes the mathematical calculations simpler. NumPy library is specifically designed for mathematical operations and is also highly optimised for the same. OpenCV-Python is a capable tool for speedy handling of computer vision issues. In our program we are making use of OpenCV version 4.1.2.

5 Results and Performance Evaluation

Our output yielded a good enhanced image which is removes blue cast and makes image more readable. The pixel range of the original image is from 0 to 254. The enhanced image is obtained by saturating 0.00001 pixels at the leftmost and rightmost sections of the original image. And then we normalized it to a scale of 0-255.

The Central Processing Unit(CPU) used in this paper is the Quad-core Intel i5, 8th generation processor with an average clock speed of 1.6 GHz which can be turbo boosted up to a maximum clock speed of 1.8 GHz.

The GPU of choice is the Tesla K80 having specifications as mentioned in Table 1 in Section 3.

Developing a parallel version of code enables us to run its application on large data (e.g. more pixels, a bigger physical model of system) in less amount of time. The result of successful parallelization is measured in terms of speedup. The speedup is nothing but the simple ratio of execution time on CPU to execution time on GPU. The speedup can be calculated by using formula as[16]:

Speed up = (Program exécution time on CPU) / (Program exécution time on GPU) (4)

In case if the serial version of an application executes in 150 seconds and the corresponding parallel version of the same application runs in 15 seconds, the speedup for the parallel application is 10 times[16]. Speedup is required when an algorithm consists of high computation. In case of CPU, it will provide high clock speed but a fewer threading functionality. While in GPU, as there are a greater number of cores a good multi-threading performance is obtained. In this algorithm normalization function is applied on each pixel co-ordinates which becomes effective if multiple pixel normalization is done parallelly.

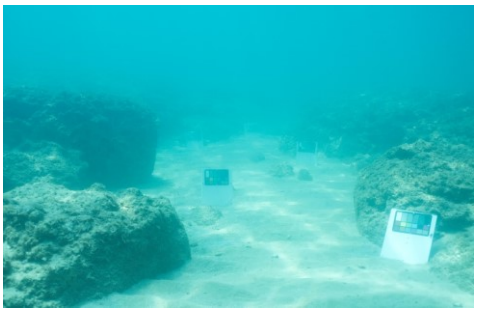

Fig. 2. Original Image.

Fig. 3. Image after undergoing the Image Enhancement.

Some of the important metrics of an image to be looked up after any kind of image processing are Mean brightness, Entropy, etc. We have found out the numerical values of this metrics and have tabulated them as follows:

Table 2. Image Metrics.

Parameters	***Numerical Values***
Program execution time in CPU	0.53 seconds
Program execution time in GPU	0.02 seconds
Speedup	26.5
Mean Brightness of original image	144.24
Mean Brightness of the enhanced image	116.45
Original image's Entropy value	6.97
Enhanced images's Entropy value	7.03

Efficiency of an image processing technique is decided on the basis of its image metrics as follows [17]:

i) Smaller the value of mean brightness better is the efficiency of the image enhancement.

ii) Bigger the value of entropy, better is the enhanced image.

As we can read from Table 2, The enhanced image obtained from applying SCB Algorithm has a low brightness value and a higher entropy as compared to the original image hence our method is fairly efficient.

6 Conclusion and Future Scope

We have provided an alternative idea for enhancing the underwater images. Our approach not necessarily improves the image quality but makes it more readable. The main aim of SCB is to balance out the hue of all the colours in the RGB spectrum and thereby reducing the dominance of blue colour in underwater images. By

using GPU, we have obtained a good speedup. Use of GPU also improves the improves the efficiency and reduces the computational time in case of a higher resolution image. By modifying the saturation percentage, we can also obtain a better enhanced image. However, it has been observed that at lower values of saturation certain artifacts start appearing in the resultant image thereby ruining the quality of the image. By keeping the saturation percentage to a suitable value, we can successfully obtain really good colour balanced images. Since the SCB algorithm is fairly simple to implement and as it is not resource intensive, a better speedup can be obtained using powerful GPUs. Our method of Underwater Image Processing using Simplest Colour Balancing Algorithm implemented on GPU is more efficient and takes less computational time as compared to other Underwater Image Processing Algorithms.

The Future Scope of our Image enhancement method is as follows:

i) SCB Algorithm can be implemented in real-time image processing.

ii) SCB Algorithm can be implemented in video processing followed by an implementation in real-time video processing.

iii) We can try to implement the Algorithm in Jetson TK1 and Jetson Nano GPUs.

References

1. K. Iqbal, R. Abdul Salam, A. Osman, A. Zawawi Talib, "Underwater image enhancement using an integrated color model" **34**, 2-12 (2007)
2. J. S. Jaffe, "Computer modeling and the design of optimal underwater imaging systems" **15**, 101–111 (1990)
3. R. Fattal, "Single image dehazing" 1–9 (2008)
4. W. Hou, D. J. Gray, A. D. Weidemann, G. R. Fournier, J. L. Forand, "Automated underwater image restoration and retrieval of related optical properties" **1**, 1889–1892 (2007)
5. N. Shamsuddin, W.F. Ahmad, B.B. Baharudin, M. Kushairi, M. Rajuddin, F. Mohd, "Significance level of image enhancement techniques for underwater images" **1**, 490-494 (2012)
6. M.S. Hitam, Wan Nural Jawahir Hj Wan Yussof, E.A. Awalludin, Z. Bachok, "Mixture contrast limited adaptive histogram equalization for underwater image enhancement" (2013)
7. S. Harsdorf, R. Reuter, S. Tonebon, "Contrast-enhanced optical imaging of submersible targets" **3821**, 378-383 (1999)
8. Y. Rzhanov, L. M. Linnett, R. Forbes, "Underwater video mosaicing for seabed mapping" **1**, 224-227 (2000)
9. R. Schettini, S. Corchs, "Underwater Image Processing: State of the Art of Restoration and Image Enhancement Methods" (2010)
10. Gonzales and Woods, *Digital Image Processing* **3,** (2008)
11. B. Daga, A. Bhute, A. Ghatol, "Implementation of Parallel Processing using NVIDIA GPU Framework" (2011)
12. E. Buber, B. Diri, "Performance Analysis and GPU vs CPU comparison for Deep Learning" (2018)
13. Z. Ying, G. Li, Y. Ren, R. Wang, W. Wang, "A new image contrast enhancement algorithm using exposure fusion framework" 36-46 (2017)
14. N. Limare, "IPOL Algorithm: Simplest Color Balance," http://www.ipol.im/pub/algo/lmps_simplest_color_balance/
15. S. Gopal Krishna Patro, Kishore Kumar Sahu, "Normalization: A Preprocessing Stage" (2015)
16. P. Patil, N. Singhaniya, C. Jage, V.A. Vyawahare, M.D. Patil, P.S.V. Nataraj, "GPU Computing of Special Mathematical Functions used in Fractional Calculus" **1**, 200-233 (2017)
17. M. Marino, M. Cozza and F. Bruno, "Evaluation of Underwater Image Enhancement Algorithms under Different Environmental Conditions" (2018)